

---

# Supplementary Material to: Weakly Supervised Dense Event Captioning in Videos

---

This Supplementary material provides proof of proposition 1, and presents more details about the model, training, and testing.

## A Proof of proposition 1(Fixed-point iteration)

**Proposition 1** (Fixed-Point-Iteration). *We define the iteration as*

$$\mathbf{S}(t+1) = l_{\theta_1}(\mathbf{V}, g_{\theta_2}(\mathbf{V}, \mathbf{S}(t))), \quad (18)$$

where  $\mathbf{S}(t)$  will converge to the fixed-point solution, i.e.  $\mathbf{S}^* = l_{\theta_1}(\mathbf{V}, g_{\theta_2}(\mathbf{V}, \mathbf{S}^*))$ , if the initial start  $\mathbf{S}(0)$  surrounds the fixed point  $\mathbf{S}^*$  sufficiently and the function  $l_{\theta_1}(\mathbf{V}, g_{\theta_2}(\mathbf{V}, \mathbf{S}))$  is locally Lipschitz continuous around  $\mathbf{S}^*$  with Lipschitz constant  $L < 1$ .

*Proof.* Since the function  $f(\mathbf{S}) = l_{\theta_1}(\mathbf{V}, g_{\theta_2}(\mathbf{V}, \mathbf{S}))$  is Lipschitz continuous with Lipschitz constant  $L \leq 1$ , we have,

$$\begin{aligned} |\mathbf{S}(t+1) - \mathbf{S}(t)| &= |f(\mathbf{S}(t)) - f(\mathbf{S}(t-1))| \\ &\leq L|\mathbf{S}(t) - \mathbf{S}(t-1)|, \\ &\vdots \\ &\leq L^t|\mathbf{S}(1) - \mathbf{S}(0)|. \end{aligned} \quad (19)$$

Since  $L < 1$ ,  $|\mathbf{S}(t+1) - \mathbf{S}(t)| \rightarrow 0$  converges to zeros as  $t \rightarrow \infty$ . Thus,  $\mathbf{S}(t)$  will converge to the fixed point that is the solution of Eq.(2) in the paper.  $\square$

## B Video Encoder & Sentence Encoder

In § 3, we have introduced the details of our model except the RNN-based video and sentence encoder used in feature extracting, which are considered not crucial in the paper. But for reproduction of our work, we will introduce them detailedly in this section.

### B.1 Video Encoder.

Reviewing § 3, the video encoder aims to encode the video frames into a set of vectors. For this purpose, a C3D[1] network is constructed to extract frame-block features and resolve short-term dependency, while a followed GRU[2] is leveraged for long-term dependency.

We denote the input video as  $\mathbf{V} = \{\mathbf{f}_t\}_{t=0}^{T_v}$  where  $T_v$  is the temporal length of the video and  $\mathbf{f}_t$  specifies the  $t$ -th frame. Following the implementation by[3], we first extract the non-overlapping C3D features from the original image frames with a interval  $\delta$ , i.e., computing  $\{\mathbf{x}_t^{(v)} = C3D(\mathbf{f}_{(t-1)*\delta+1} : \mathbf{f}_{t*\delta})\}_{t=0}^{T_v/\delta}$ , where  $\delta = 16$ .

The extracted C3D features are fed into a GRU network to learn the long-term dependence. Specially, we set the initial hidden state as zero, i.e.  $\mathbf{h}_0^{(v)} = \mathbf{0}$ , and compute the following outputs and hidden states recursively by

$$\mathbf{v}_t, \mathbf{h}_t^{(v)} = GRU_{video}(\mathbf{x}_t^{(v)}, \mathbf{h}_{t-1}^{(v)}). \quad (20)$$

The output sequence  $\{\mathbf{v}_t\}_{t=0}^{T_v/\delta}$  and hidden states  $\{\mathbf{h}_t^{(v)}\}_{t=0}^{T_v/\delta}$  are used in the followed processes of sentence localizing and caption generating as video representation (In the original paper, we omit  $\delta$  and denote  $T_v/\delta$  as  $T_v$  and the extract feature as  $\{\mathbf{v}_t\}_{t=0}^{T_v}$  for simplify).

## B.2 Sentence Encoder

Similar to the video encoder, sentence encoder aims to encode natural sentences into vector representations. Given a sentence  $\mathcal{C} = \{\mathbf{w}_t\}_{t=0}^{T_c}$ ,  $\mathbf{w}_t \in \{0, 1\}^V$  denote the one-hot encoding of the  $t$ -th word, where  $V$  is the vocabulary size. Then,  $\mathbf{w}_t$  is embedded into a  $k$ -dimensional vector by  $\mathbf{x}_t^{(c)} = \mathbf{W}_e \mathbf{w}_t$ , with  $\mathbf{W}_e \in \mathcal{R}^{k \times V}$  being the trainable embedding matrix. Similar to the video encoder, the embedded features  $\{\mathbf{x}_t^{(c)}\}_{t=0}^{T_c}$  are fed into a GRU model. Specially, the initial hidden state  $\mathbf{h}_0^{(c)}$  is set to be zero, and the outputs and following hidden states are computed as:

$$\mathbf{c}_t, \mathbf{h}_t^{(c)} = GRU_{text}(\mathbf{x}_t^{(c)}, \mathbf{h}_{t-1}^{(c)}). \quad (21)$$

After the encoding stage, the final outputs  $\{\mathbf{c}_t\}_{t=0}^{T_c}$  and hidden states  $\{\mathbf{h}_t^{(c)}\}_{t=0}^{T_c}$  are used in the followed networks.

## B.3 More Details

**I.** As both **Sentence Localizer**  $l_{\theta_1}$  and **Caption Generator**  $g_{\theta_2}$  use the video features for further processing, they can choose whether to share the same video encoder parameters or not. In our previous experiments, we found that both strategies reach similar performances. The reported results are obtained using different video encoder parameters.

**II.** Also, as training a C3D network from scratch is very time-consuming and error-prone, we directly adopt the public-available C3D features<sup>1</sup>. The public-available features are 500-way features reduced from the original 4096-way output of C3D’s fc7 layer using PCA. In our experiments, the features are denoted as  $\{\mathbf{x}_t^{(v)}\}_{t=0}^{T_v/\delta}$  and fed into the  $GRU_{video}$  for further processing.

# C Training & Testing Details

## C.1 Training

Because our model consists of two submodels and several term losses, there may be several ways to train the whole model. In this section, we introduce the strategy we used in our experiments. Firstly, the model is pretrained with our predefined *Fake Proposal*, i.e.  $\mathbf{S}^{(f)} = (0.5, 1)$ . After pretraining for several rounds, we train the proposed three term losses alternatively. Details are shown in Algorithm1.

## C.2 Testing

As our model is not directly trained on the dense event captioning problem, we provide an extra explanation on the strategy we used in testing. In short, our training strategy and losses force the model meets the requirement of fixed-point iteration, so we use a random bunch of segments  $\{\mathbf{S}_i^{(r)}\}_i^{N_r}$  as initial temporal segments, feed them into the cycle system  $\mathbf{S}'_i = l_{\theta_1}(\mathbf{V}, g_{\theta_2}(\mathbf{V}, \mathbf{S}_i^{(r)}))$ . Considering that those random segments will converge to the true event temporal segments or diverge to some unknown random segment, the distance between  $\mathbf{S}'_i$  and  $\mathbf{S}_i^{(r)}$  should be small if they are in the neighborhood of a certain event. Also, we measure the distance between  $\mathbf{S}'_i$  and  $\mathbf{S}'_j$  to further reduce redundancy(This step is not critical, but can reduce the repetition of temporal segments.). Specifically:

- if  $dist(\mathbf{S}'_i, \mathbf{S}_i^{(r)}) > \Theta_1$ , remove  $\mathbf{S}'_i$  from the predicted temporal segment set

<sup>1</sup><http://activity-net.org/challenges/2016/download.html#c3d>

- if  $\text{dist}(\mathbf{S}'_i, \mathbf{S}'_j) < \Theta_2$ , merge  $\mathbf{S}'_i, \mathbf{S}'_j$  as:  $\mathbf{S}''_i = \text{union}(\mathbf{S}'_i, \mathbf{S}'_j)$

where  $\text{dist}(\cdot)$  computes the IoU between  $\mathbf{S}_1, \mathbf{S}_2$ :

$$\text{dist}(\mathbf{S}_1, \mathbf{S}_2) = \frac{\text{intersection}(\mathbf{S}_1, \mathbf{S}_2)}{\text{union}(\mathbf{S}_1, \mathbf{S}_2)} \quad (22)$$

Details about testing are shown in Algorithm2.

---

**Algorithm 1** Training pipeline for the WS-DEC model

---

**Input:**  $\mathcal{D}$  // the dataset iterator for training  
**Input:**  $l_{\theta_1}, g_{\theta_2}$  // the random initialized sentence localizer, caption generator  
**Input:**  $\{\mathbf{S}_j^{(a)}\}_{j=0}^{N_a}$  // anchor segments for training  $l_{\theta_1}$   
**Input:**  $\mathbf{S}^{(f)}$  //  $\mathbf{S}^{(f)} = (m^{(f)}, w^{(f)})$  is used to pretrain the caption generator  
**Output:**  $\theta_1, \theta_2$  // trained parameters for sentence localizer and caption generator

- 1:  $step \leftarrow 0$
- 2: **while**  $step < \text{pretrain\_step}$  **do** // we pretrain the model with fake proposal
- 3:   **for**  $(\mathbf{V}, \{\mathbf{C}i\}_{i=0}^{N_v}) \in \mathcal{D}$  **do**
- 4:      $\mathbf{C} \leftarrow \text{RANDOMCHOOSE}(\{\mathbf{C}i\}_{i=0}^{N_v})$  // randomly choose a sentence
- 5:      $\mathbf{C}' \leftarrow g_{\theta_2}(\mathbf{V}, \mathbf{S}^{(f)})$  // obtain the fake generation
- 6:      $\mathcal{L}_c \leftarrow \text{dist}(\mathbf{C}, \mathbf{C}')$  // compute the loss
- 7:      $\theta_2 \leftarrow \theta_2 - \eta_2 \frac{\partial \mathcal{L}_c}{\partial \theta_2}$  // update parameters with SGD
- 8:   **end for**
- 9:    $step \leftarrow step + 1$
- 10: **end while**
- 11: **while**  $step < \text{training\_step}$  **do** // we train the model with three term losses alternatively
- 12:   **if** training  $L_c$  and  $L_s$  **then** // train the model with  $\mathcal{L}_c$  and  $\mathcal{L}_s$
- 13:     **for**  $(\mathbf{V}, \{\mathbf{C}i\}_{i=0}^{N_v}) \in \mathcal{D}$  **do**
- 14:        $\mathbf{C} \leftarrow \text{RANDOMCHOOSE}(\{\mathbf{C}i\}_{i=0}^{N_v})$
- 15:        $\mathbf{S}' \leftarrow l_{\theta_1}(\mathbf{V}, \mathbf{C})$
- 16:        $\mathbf{C}' \leftarrow g_{\theta_2}(\mathbf{V}, \mathbf{S}' + \delta)$
- 17:        $\mathbf{S}'' \leftarrow g_{\theta_2}(\mathbf{V}, \mathbf{C}')$
- 18:        $\mathcal{L} \leftarrow \text{dist}(\mathbf{C}, \mathbf{C}') + \text{dist}(\mathbf{S}', \mathbf{S}'')$
- 19:        $\theta_1 \leftarrow \theta_1 - \eta_{c1} \frac{\partial \mathcal{L}}{\partial \theta_1}$
- 20:        $\theta_2 \leftarrow \theta_2 - \eta_{c2} \frac{\partial \mathcal{L}}{\partial \theta_2}$
- 21:     **end for**
- 22:   **end if**
- 23:   **if** training  $L_a$  **then** // train the model with  $\mathcal{L}_a$
- 24:     **for**  $(\mathbf{V}, \{\mathbf{C}i\}_{i=0}^{N_v}) \in \mathcal{D}$  **do**
- 25:        $\mathbf{C} \leftarrow \text{RANDOMCHOOSE}(\{\mathbf{C}i\}_{i=0}^{N_v})$
- 26:        $\{\mathbf{C}j^{(a)}\}_{j=0}^{N_a} \leftarrow \{g_{\theta_2}(\mathbf{V}, \mathbf{S}_j^{(a)})\}_{j=0}^{N_a}$  // anchor segments  $\rightarrow$  anchor sentences
- 27:        $\text{confidence} \leftarrow l_{\theta_1}(\mathbf{V}, \mathbf{C})$  // confidence about each anchor regarding  $\mathbf{V}$  and  $\mathbf{C}$
- 28:        $\{\text{score}_j^{(a)}\}_{j=0}^{N_a} \leftarrow \{\text{METEOR}(\mathbf{C}, \mathbf{C}j^{(a)})\}_{j=0}^{N_a}$  // compute Meteor score
- 29:        $\text{label} \leftarrow \arg \max_j \{\text{score}_j^{(a)}\}_{j=0}^{N_a}$  // the one with highest score is considered as label.
- 30:        $\mathcal{L}_a \leftarrow \text{CROSSENTROPY}(\text{confidence}, \text{label})$
- 31:        $\theta_1 \leftarrow \theta_1 - \eta_{a1} \frac{\partial \mathcal{L}_a}{\partial \theta_1}$  // cross entropy loss
- 32:     **end for**
- 33:   **end if**
- 34:    $step \leftarrow step + 1$
- 35: **end while**
- 36: **return**  $\theta_1, \theta_2$

---

---

**Algorithm 2** Testing pipeline for the WS-DEC model

---

**Input:**  $\mathbf{V}$  // a video for testing(never seen in the training set)  
**Input:**  $l_{\theta_1}, g_{\theta_2}$  // the learned sentence localizer, caption generator  
**Input:**  $\{\mathbf{S}_i^{(r)}\}_{i=0}^{N_r}$  // initial random bunch temporal segments(fixed for all testing example)  
**Output:**  $\{\mathbf{C}i, \mathbf{S}_i\}_{i=0}^{N_e^{(v)}}$  // the predicted events within the video

- 1:  $step \leftarrow 0$  // iteration round
- 2:  $N_e^{(v)} \leftarrow N_r$  // initial number of event equals the predefined one
- 3:  $\{\mathbf{S}_i\}_{i=0}^{N_e^{(v)}} \leftarrow \{\mathbf{S}_i^{(r)}\}_{i=0}^{N_r}$  // initial temporal segment equals the predefined temporal segments
- 4:  $\{\mathbf{C}i\}_{i=0}^{N_e^{(v)}} \leftarrow \{g_{\theta_2}(\mathbf{V}, \mathbf{S}_i)\}_{i=0}^{N_e^{(v)}}$  // generate initial description
- 5: **while**  $step < iteration\_step$  **do** // iterate until max iteration step
- 6:  $\{\mathbf{S}'_i\}_{i=0}^{N_e^{(v)}} \leftarrow \{l_{\theta_1}(\mathbf{V}, \mathbf{C}i)\}_{i=0}^{N_e^{(v)}}$  // re localize the generated sentence
- 7:  $\{\mathbf{S}''_i\}_{i=0}^{N_e^{(v)'}}$   $\leftarrow$  SEGMENTMERGE( $\{\mathbf{S}_i\}_{i=0}^{N_e^{(v)}}$ ,  $\{\mathbf{S}'_i\}_{i=0}^{N_e^{(v)}}$ ) // merge segments based on § 4.3.3
- 8:  $N_e^{(v)} \leftarrow N_e^{(v)'}$  // update the predicted number of events
- 9:  $\{\mathbf{S}_i\}_{i=0}^{N_e^{(v)}} \leftarrow \{\mathbf{S}''_i\}_{i=0}^{N_e^{(v)'}}$  // update events temporal segments
- 10:  $\{\mathbf{C}i\}_{i=0}^{N_e^{(v)}} \leftarrow \{g_{\theta_2}(\mathbf{V}, \mathbf{S}_i)\}_{i=0}^{N_e^{(v)}}$  // update description sentences
- 11:  $step \leftarrow step + 1$
- 12: **end while**
- 13: **return**  $\{\mathbf{C}i, \mathbf{S}_i\}_{i=0}^{N_e^{(v)}}$

---

## References

- [1] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.
- [2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [3] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 706–715, 2017.